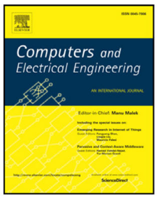
Computers and Electrical Engineering 65 (2018) 18–31

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



A novel Wireless Network-on-Chip architecture with distributed directories for faster execution and minimal energy[☆]

Kishore K. Chidella[*], Abu Asaduzzaman

Wichita State University, 1845 Fairmount st.; Jabara Hall 256, Wichita, Wichita, KS 67260, United States

ARTICLE INFO

Article history:
Received 28 February 2017
Revised 20 December 2017
Accepted 21 December 2017

Keywords:
Distributed directories
Mesh architecture
Multicore architecture
Performance analysis
Wireless Network-on-Chip

ABSTRACT

Wireless Network-on-Chip (WNoC) architectures are introduced to improve performance by reducing the core-to-core communication latency. Conventional WNoCs broadcast messages that increase bandwidth-traffic, communication delay, and power consumption. Studies show that directory-based architectures have potential to address message broadcasting and improve performance. This work proposes a novel WNoC architecture with distributed directories (WNoC-DDs) that supports wireless communications to enhance faster execution by reducing latency. VisualSim software package is used to model and simulate the proposed WNoC-DDs, a WNoC with centralized directory (WNoC-CD), and a traditional 2D mesh by processing different communication scenarios. The proposed architecture helps reduce the total hop count and unwanted broadcasting among nodes in a WNoC-DDs. Experimental results show that the proposed WNoC-DDs reduces communication delay up to 20.54% and 5.40%, respectively, when compared to mesh and WNoC-CD. Similarly, the proposed WNoC-DDs reduces power consumption up to 73.56% and 19.97%, respectively, when compared to mesh and WNoC-CD.

1. Introduction

The dominant technology such as Network-on-Chip (NoC) is becoming trendy and can solve performance limitations of traditional wired interconnects and productive for System-on-Chip (SoC) architectures. Recent studies indicate that lot of products, such as, processors, cell phones, memory subsystems and many other embedded products are integrated on a single chip and interconnected by NoC [1–3]. The design of multicore systems makes easy to solve complex jobs by working concurrently in parallel with improved execution speed and reduced power consumption [4,5]. Multithreading is a process in which a central processing unit (CPU) can execute several number of threads simultaneously. Memory-balanced scheduling is a thread scheduling approach that improves the performance by balancing memory access requirements but at the cost of interconnects width and bandwidth [6]. However, the programming for large scale multicore architecture is always challenging [7]. The functionality of multicore can be outstanding when the cache coherence is reduced, and it could be less in private memory multicore architectures but are expensive. The shared memory plays a trade-off approach of cost and cache coherence problem. Snooping protocols address coherence issues but are limited to small core counts, whereas

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. A. Isazaheh.
* Corresponding author.
E-mail addresses: kkchidella@shockers.wichita.edu (K.K. Chidella), abu.asaduzzaman@wichita.edu (A. Asaduzzaman).

https://doi.org/10.1016/j.compeleceng.2017.12.038
0045-7906/© 2017 Elsevier Ltd. All rights reserved.

directory protocols are good for large cores. Recent studies indicate that the snoopy protocols can be extended to 36-core but the power and area are the major drawbacks [8]. The performance and efficiency of communication among cores can be achieved with the implementation of additional parallelism and multithreading.

Multicore designs should assure less chip area, reduced communication latency and reduced power consumption thereby, providing better communication among cores on the chip. NoC architecture, is a technology proposed to overcome the problem of large communication delay among cores in a multicore architecture. Multiple interconnects are proposed to address latency and power issues such as concentrated-sparse mesh [9], millimeter wave wireless interconnects [10], crossbar on-chip interconnects [11]. The primary purpose of those interconnects is to overcome power and latency issues. But the designs are still tough to address performance issues within low cost and scalability. However, with the constraints and limitations of multicore designs, the development of efficient NoC grabbed an outstanding attraction.

The routing paths for efficient communication among cores are different and follow adaptive or non-adaptive algorithms in multicore architectures. In traditional mesh multicasting, the popular non-adaptive technique is XY routing algorithm [12]. Wireless Network-on-Chip with centralized directory (WNoC-CD) architecture [13] uses an adaptive XY routing algorithm to decide the path between nodes [14]. Also, WNoC-CD uses the buffer management to improve the performance by taking care of queuing delays without affecting the throughput rate [15]. The topology of a multicore architecture on chip plays a predominant role in the communication delay. The introduction of directory in multicore architectures, can improve the performance like faster execution and overcome the cache coherence problems [16–18]. However, the centralized directory lacks its performance and slows connections for several reasons such as insufficient bandwidth, increase in network size, and heavy traffic [19].

The existing interconnection technologies such as RF-I and UWB have speed, bandwidth, area, RC (Resistor Capacitor) wired interconnect, and power issues. The proposed distributed directories are Stanford Directory Architecture for Shared (DASH) memory that addresses several issues such as speed problem, bandwidth, traffic, area, power consumption and data sync. In detail, speed problem can be reduced by using XY routing algorithm, bandwidth as well as traffic issues reduced by distributed directories mechanism, area can be narrowed by reducing RC interconnects, and the power is reduced on factors such as selection of shortest path to reach destination. Data sync is better with distributed directories compared to synchronization from individual cores level. In this paper, we propose distributed directories based architecture with wireless routers to overcome centralized directory limitations such as reducing communication delay, hop count, and power consumption.

Section 2 summarizes related published articles. In Section 3, the proposed distributed directories based multicore architecture with wireless routers is introduced. In Section 4, the experimental details are described. In Section 5, experimental results and related discussions are presented. In Section 6, the conclusions of the work are presented.

2. Background study

In this section, we discuss some popular network topologies used in WNoC to reduce the performance bottleneck that may reduce the data throughput, and scalability. We also consider the issues of cache coherence that lead to complication of data exchange between cores, and how a Stanford DASH architecture will address the coherence using customized MESI protocol.

2.1. Popular network topologies

Mostly, the multicore architectures are designed to enhance the performance by utilizing several cores for multiple tasks in parallel. However, the communication between cores is the most influenced consideration to reduce the performance bottlenecks such as latency, power consumption, area, and throughput. The popular interconnect topologies that are employed in NoC architectures are bus, ring, crossbar, and mesh. Photonic integrated NoC overcome the bandwidth barriers in traditional or electronic NoC. In photonic integrated NoC, bus topology has high latency when compared to ring topology for higher loads. Even though the bus topology provides the shortest path, it requires additional transceivers which increases cost and power consumption. Unidirectional ring topology provide better scheduling performance but requires more wavelength and large packet length [20]. Bit error rate is slightly more in bus topology when compared to ring topology. However, the performance of bus and ring topologies is worst for concurrent transmissions of same wavelength that can cause cross talk [21]. Modular decoupled cross bar architecture is efficient for area and power consumption but has power gating, excessive wiring, and performance issues based on workload [22]. Mesh is a popular network topology for multicore architectures that can handle high volume traffic. In a 2D (two-dimensional) mesh network, all cores are connected in a crossbar connection. Mesh topology is introduced in NoC because of its simplicity and scalability. Sparse mesh interconnects are advanced to traditional mesh solving bandwidth and performance issues in high traffic. However, the concentrated-sparse mesh network latency is higher for low traffic workloads [9]. Even though, the mesh has better advantages compared to bus, ring, and crossbar topologies, the issues of bandwidth and inadequate performance yet to be addressed.

Nowadays, the cores are increasing tremendously on a single chip and so it is essential to reduce the wired connects. Wired connects brings several issues such as latency, area, and traffic between the cores. Therefore, the adaption of wireless interconnects plays a crucial role to address the complications of wired interconnects. A complete setup of wireless interconnects is not satisfactory due to bandwidth issues and so the hybrid integration of wired and wireless interconnects typically

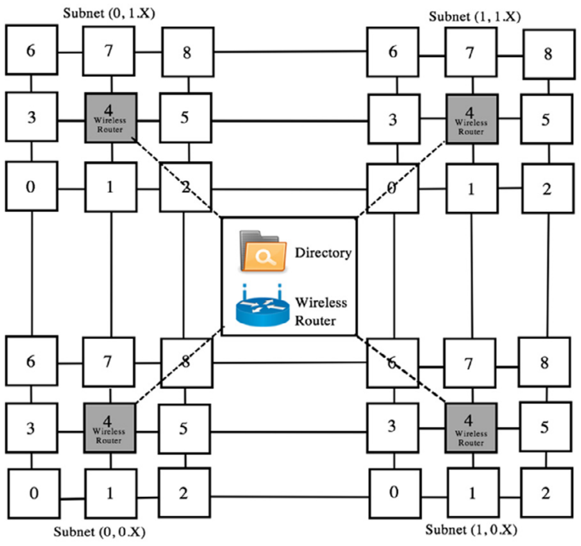


Fig. 1. WNoC-CD with wireless routers.

plays an upper hand topology. Traditional WNoC architecture, which is designed on top of traditional mesh topology can handle the issues of network congestion and power consumption.

2.2. WNoC architecture

Multiple processor cores are integrated on a single chip that leads to Chip Multiprocessor Network-on-Chip (CMP NoC). We considered a 2D WNoC that has 36-cores, which is divided into 4 subnets and each subnet has 9-cores. When compared to 2D, the 3D (three-dimensional) NoCs have advantages from 3D integrated circuits (ICs) and NoCs thereby, providing improvements in latency and power consumption [23]. However, 3D NoCs have shortcomings such as thermal stability, failure of concurrent communication in third dimension, and high chances of contention and blocking [24].

To overcome the scalability issues, considering the 3D NoC shortcomings hybrid communication on 2D NoC that has wired and wireless interconnects is preferred. In WNoC architecture, network based processor array (NePA) is the primary component which takes care of data transfer processing and routing between cores. Each core in WNoC is a processing element (PE) that has network interface (NI), and a router. Each PE has its own execution unit, program memory, and processing core that takes the responsibility in accomplishing the data transfer between the cores. NI allows the cores connected to the network to exchange data or any other information. NI sends the commands to router through internal port, after receiving the instructions from PE. Each router has 4 links, that is East, West, North, and South. Router directs the packets based on the address of destination and is directly connected to its 4 neighbor cores through links. The router job of each core is to send the packets to its neighbor core only, and thus it is the responsibility of each router to forward the data between neighboring routers. However, the control on packets from source to destination is managed by NePA using adaptive XY routing algorithm. If the ports or routers are busy, then the algorithm uses buffer utilization technique and selects an alternative path without adding the latency or waiting in a queue.

2.3. WNoC with centralized directory

Wireless routers are capable of transferring packets via wired as well as wireless interconnects. In WNoC-CD, every subnet center core (say, core-4) is a hybrid router that has wired links to its directly connected subnet cores and wireless links to other wireless routers in different subnets. Therefore, WNoC-CD is capable of transferring packets through wired and wireless links and is shown in Fig. 1 [13]. The centralized directory updates data for every change in subnets and tracks the network traffic [25]. In WNoC-CD, processing cores are divided into various subnets which have one wireless router responsible for providing wireless communication for the other subnets. Each subnet has address in X value and Y value. The address is like the coordinates representation in a XY graph. For example, from the origin of subnet (0, 0.x): ‘0’ indicates zero point on X-axis, and from ‘0.x’, ‘0’ represents the zero point on Y-axis; x indicates the specific core in that subnet, which have 9 cores (numbered as, 0 to 8).

The directory is centralized and keeps track of each subnet using customized MESI protocol. The MESI protocol [26] helps in reducing the cache coherence by updating the status, for the changes occurred in cores through directory to other cores. The features of addressing a specific core in a network helps WNoC-CD provide much faster routing decisions as well as a scalable hierarchical system. In short, it’s a hybrid combination of WNoC and DASH architectures. The DASH architecture

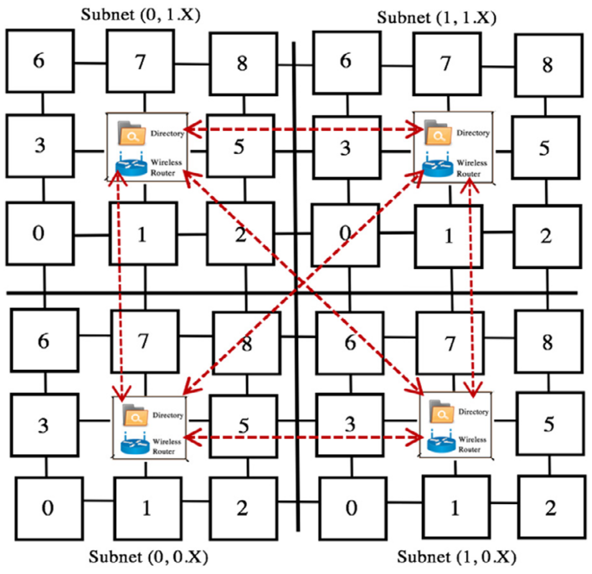


Fig. 2. Proposed architecture with distributed directories and wireless routers.

enhances the performance in shared memory where the cores are combined into a cluster and work together to accomplish the given job [27,28]. To save the bandwidth and reduce broadcasting issues of traditional NoC, the DASH architecture uses message passing technique to communicate between cluster to cluster communication. The minimal adaptive routing algorithm delivers shortest path and the directory maintains data sync among the subnets.

3. Proposed distributed directories for WNoC

To improvise the performance of WNoC architecture, distributed directories are introduced into this work that can manage data sync of all subnets, also maintaining minimal routing path, which in further allows faster execution with minimal energy. The proposed architecture is a hybrid combination of the traditional WNoC with distributed directories and DASH architecture. The major goal of the proposed multicore architecture is to reduce the communication delay among the cores by decreasing the number of hops required to travel from a source node to a destination node using the distributed directories and wireless routers. The key design considerations include: Partitioning cores into subnets, designing directories, directories working principle, and communication between in and out-subnets using directories. The major advantage of the distributed directories is performing the data sync by broadcasting the updates to all other directories without any waiting time, unlike centralized directory.

3.1. Partitioning cores into subnets

Considering the WNoC and DASH architectures, the proposed architecture divides the cores on the die into clusters, called subnets, as shown in Fig. 2. In every subnet, Core-4 in Fig. 2 contains a wireless router and a directory. Considering a 6×6 mesh topology, the cores are structured into 3×3-core subnets, forming four quadrants. The dotted line represents the wireless connections with the other subnets center core and they are connected to one another. Each center core is designed with a wireless router and a directory, which is introduced as the proposed directory. All the cores inside a subnet are local to the subnet and the cores outside of a subnet are remote cores for that subnet. A source core places its request for the data on the bus and if the data is not found among the caches of all the cores in the subnet, then a request is sent by its subnet wireless router with directory to the subnet destination directory for the requested block of data. The directory has enough memory to fetch the data between cores and they are responsible for the synchronization of data without indulging the cores, and thus the cores are relaxed.

3.2. Designing distributed directories

The directory contains the information of all other subnets that includes data sync, minimal routing path. The directory is integrated with wireless router in the central core (Core-4) of each subnet. The proposed architecture has distributed directories, where all directories are identical with equal priority. Each core is having its own cache, and when it requires any data then it tries to pull from the cache. If the cache is not having the required data then it sends a request to its subnet directory. The request from an individual core is random and the directory function is dynamic to handle the requests. The

Table 1
System parameters.

System parameters	Relevant value
Cache size/core	1 KB
Each cache block size	128 Bytes
Number of cache blocks/core	8
Number of entries/ 9-core	81
Number of entries/ 36-core	324

cache size of each core is split into cache block/cache line. When the directory receives the request from core for data, then the directory check the block individually which is defined as entry. The status of every subnet before and after the requests of data with-in or out of subnets is controlled and monitored by the directory. The number of entries required for a request depends on the cache size and block size. The parameters for a cache size, block size, number of entries is listed in Table 1.

3.3. Working principle of distributed directories

With the design of distributed directories, the pressure of accomplishing tasks on each directory is reduced unlike centralized directory. In the proposed system, customized MESI protocol is used. The directories with the help of customized MESI protocol can sync data or exchange information between directories. The directories will take care of data synchronization and thus the local cores of every subnet are free from the role of synchronization. Each directory is responsible to update their individual status and data sync information to other directories. The cores are responsible to send the information to neighbors only, which are one hop distance. Whenever, the information reaches the directory, then the directory performs the necessary operations such as sending the data to destination core or requesting data from any selective core. The data in and out from cores, as well as read or write data into block memory of a core is handled by the directories only.

Basically, for any task a core may read the data from another core or write the data to another core, apart from their own core activities. The cache size of 1 KB has 8 blocks, where each block is 128 Bytes. For example, if core-5 does not find any required data from its cache then it requests a cached block from the main memory and the details of cached block copy is also stored in the directory. The directories are then synced to maintain data consistency.

For any core, initially all the blocks are empty and the status is represented as ‘0’. Suppose if the data is to be fetched from 6th block of its own subnet or another subnet, then the status of that 6th block is changed to Exclusive (E). ‘E’ represents the copy is clean and it is existed in one cache. If the same block is fetched again, then the status remains unchanged. When the core-5 performs a write operation in a block, then the status is changed to Modified (M). ‘M’ represents the cache copy is dirty and it is changed only in its cache. To spare other cores from the dirty copy, an Invalidate (I) copy is sent to other cached blocks to inform the cache is not valid anymore. If any block is shared between multiple cores, then the status is changed to Shared (S). The directory tracks the information of each core and updates other directories accordingly to get rid of data inconsistency.

In centralized directory based architecture, synchronization is complicated as it must put the requests from other cores in a queue and they are updated in a serial passion. This indicates traffic congestion and demand of bandwidth with larger capacity of directory, which makes us to think about optimizing the drawbacks of centralized directory. The proposed distributed directories use broadcasting technique to update/sync the data that resolves the drawbacks of centralized directory. However, the proposed has slightly risen in power consumption only within subnet cases, but has reduced determining route time and hop counts for all other subnet cases.

3.4. In-Subnet and out-subnet communication using distributed directories

Every communication between cores is established through its individual directory in a subnet. The communication among cores inside a subnet follows mesh principle and so the delay in all the three architectures is uniform and they go through the wired network. The directory is updated on every task individually. To communicate with cores in different subnets, directory and wireless routers are used. A directory is implemented as a special powerful core and is shown in Fig. 2 (Core-4 of each subnet). The directory quickly provides information regarding the status and address of a block cached by cores (if any). The cores association is essential to improve communication excellence. If the destination core is physically ≤ 1 hop then the cores go through wired link and finally update the directory. As a result, the communication delay among the cores is reduced significantly; this is because the source core gets the information about the destination core (i.e., requested data) quickly. In this work, we evaluate the performance of each architecture by calculating the communication delay, hop count, and power consumption. The time taken for the packets transmitted from source to destination node is described as communication delay. Intermediate cores lead to minimum delay whereas the destination node results in large delay due to the packet processing. The total hop count of any task between source to destination node is evaluated by summing the hops for communication setup (forward path, return path), and hops for fetching data from destination to source node. Power consumption is calculated by determining the routes, identifying wired and wireless links, subnets or core average networks based on the architecture, number of directories involved in accomplishing the task.

Table 2
Assumptions for calculating hop count and communication delay.

Parameters	Relevant value
Packet length	64 bit
Heaer flit size	8 Bytes
Entire Packet size	80 Bytes
Delay due to intermediate cores	4 units
Delay due to destination core	40 units
Delay between directly connected cores	1 unit
Delay between wireless router cores	1 unit

4. Experimental details

The proposed WNoC architecture with distributed directories is evaluated and compared with traditional mesh and WNoC-CD architectures. To model and simulate our proposed system, we use VisualSim Architect 15 tool [29]. The tool is popular for designing computation systems, which provides the flexibility of modeling and simulating customized designs. The tool is efficient to analyze performance trade-offs between various architectures using bandwidth utilization, communication delay, routing metrics, and power consumption.

4.1. Assumptions

The characteristics of all three architectures with wired communication are assumed as of unique behavior. WNoC-CD and WNoC-DDs architectures are basically built on top of traditional mesh architectures. However, there are few changes in determining the routes and fetching data between cores. We considered 6×6 cores layout as common for all the three architectures. The performance of architectures can be compared by providing a common workload and it is generated by VisualSim Architect 15. The data exchange between cores is in the form of packets. The proposed system uses wormhole packet switching to transfer data between cores that requires less buffer size and less silicon area which eventually reduces power consumption. To reduce the buffer size, each packet is divided into flow control units commonly known as flits. Each packet has a header flit, payload/data flit, and tai flit. The header flit reserves the buffer and routes other flits to reach the destination core by using virtual channels. The tail flit of each packet terminates the reserved buffer and thus allowing other flits to use that buffer. In few tasks, the number of cores between source and destination can be multiple and they are specified as intermediate cores. The intermediate cores do not process the whole packet, instead they consider only the header flit to confirm the core address and then forwards to other core if the address is not matched to its core. In this way, the delay due to intermediate cores can be reduced compared to destination core. However, the destination core should process the whole packet received in flits and so the delay is larger compared to intermediate cores. In the proposed system, distributed directories are introduced and so the directory update is essential to maintain data synchronization. The assumptions for calculating hop count and communication latency are listed in Table 2.

4.2. Tasks at hand

There are many possible ways of communication between cores in a 36-core architecture. Each possible communication for data exchange between nodes is considered as task. In this work, we examined different tasks that has in-subnet and out-subnet scenarios. To observe the performance of three architectures in worst and best conditions, random sequence of events as tasks is generated. Tasks 1 to 20 covers the out-subnet scenarios as well as Tasks 21 to 25 includes in-subnet scenarios. VisualSim Architect 15 generates the workload sequentially that covers the random requests of subnets and it is illustrated in Table 3.

4.3. Determining route between source and destination nodes

In all the three architectures, the cores communicate and perform read/write operations as required. The source core will reach the destination by following the routing technique which depends on its architecture. Adaptive XY routing algorithm is used in all the three architectures we considered. As we discussed before, the communication between cores depends on the type of interconnect topology and so the data transfer time varies with respect to its pattern or layout. For example, the mesh architecture uses multicasting technique to communicate between cores. In mesh architecture, the source core will check the status (busy/idle) of destination by sending a request signal. The destination core will acknowledge its status to source core. The total time/path taken for request and acknowledge is considered as communication setup time, which is avoided in WNoC directory architectures. The directory is a coprocessor that has control logic, data storage, and linked to router that has wired and wireless capability. In mesh architecture, communication delay increases as the number of intermediate cores increases, and thus the power consumption also increases. The directory overcomes the delay, as they follow a route on subnet basis with the introduction of wireless routers. However, centralized directory architecture need more hops than proposed WNoC-DDs. The introduction of individual directories in each subnet improves the performance

Table 3
Source and destination nodes for different communication tasks.

Different scenarios	Source Node (S)	Destination Node (D)	Subnet Location
Task 1	Core (0, 0.0)	Core (1, 1.8)	Different
Task 2	Core (0, 0.4)	Core (1, 1.4)	Different
Task 3	Core (0, 0.7)	Core (1, 0.1)	Different
Task 4	Core (0, 0.3)	Core (0, 1.5)	Different
Task 5	Core (1, 0.5)	Core (0, 1.2)	Different
Task 6	Core (1, 0.7)	Core (0, 1.5)	Different
Task 7	Core (0, 1.0)	Core (1, 0.0)	Different
Task 8	Core (0, 0.8)	Core (1, 1.6)	Different
Task 9	Core (0, 0.7)	Core (0, 1.1)	Different
Task 10	Core (1, 1.5)	Core (1, 0.2)	Different
Task 11	Core (0, 1.3)	Core (0, 0.1)	Different
Task 12	Core (0, 1.4)	Core (1, 0.6)	Different
Task 13	Core (1, 0.1)	Core (1, 1.1)	Different
Task 14	Core (1, 1.2)	Core (0, 0.8)	Different
Task 15	Core (1, 0.6)	Core (0, 1.2)	Different
Task 16	Core (1, 0.4)	Core (0, 1.7)	Different
Task 17	Core (1, 1.3)	Core (0, 1.3)	Different
Task 18	Core (0, 1.2)	Core (1, 1.0)	Different
Task 19	Core (0, 0.1)	Core (1, 0.7)	Different
Task 20	Core (1, 0.2)	Core (0, 1.6)	Different
Task 21	Core (0, 0.6)	Core (0, 0.5)	Same
Task 22	Core (1, 0.7)	Core (1, 0.8)	Same
Task 23	Core (0, 1.4)	Core (0, 1.2)	Same
Task 24	Core (1, 1.6)	Core (1, 1.2)	Same
Task 25	Core (0, 1.7)	Core (0, 1.1)	Same

of proposed WNoC-DDs architecture. If the destination is only one hop distance, then all the networks behave as mesh and thus have the same communication delay. However, the routing path to communicate within the subnet is same and so the delay is also unique for all three architectures. In most scenarios, the proposed system takes less time than the centralized directory and mesh architectures.

The proposed architecture is faster because the traffic for communicating between subnets is carried through directories whereas in centralized directory based architecture, the request is from wireless routers. The directory to directory communication is more adequate and faster as they send a request in indexed search format to other. The core in each subnet will get the required information from its own subnet directory and thus the traffic is limited to 8 cores only excluding directory core. The directory uses a point-to-point message passing technique that minimizes the bandwidth and broadcasting issues. The directory maintains the status of all cores (36-core) and to accomplish it every individual directory updates their subnet activities to other subnets using customized MESI protocol.

Let's consider the Task 1, which involves the end-to-end distance between source and destination nodes. The information is generally transmitted in packets that have header, payload and trailer. Here the header size, say 8 bytes and the whole packet is 80 bytes. Therefore, if the delay due to an intermediate core is 4 units, the delay caused due to a destination core is assumed to be 40 units. The intermediate cores check only the header flit and so each intermediate core causes 4 units of delay. In mesh, for the end-to-end task, they are 9 intermediate cores and one destination core excluding source core. So, delay due to 9 intermediate cores will be 36 ($4*9 = 36$) units and the destination core takes 40 units, which will make the total as 76 units. In WNoC-CD, the centralized directory is considered as destination core. So, in the task, it has 2 intermediate cores and 1 destination core (centralized directory) involved. In detail, delay due to intermediate cores is 8($4*2 = 8$) units and the destination core takes 40 units, which will make the total as 48 units. In the proposed WNoC-DDs, the individual directory is considered as destination. So, in the task, it has only one intermediate core that takes 4 units and one destination (directory) core that takes 40 units, which will make the total as 44 units.

4.4. Hop count

The link between two cores is determined as one hop distance. WNoC with centralized and distributed architectures are built on top of mesh architecture, the hop count (HC) is only one when the connected links between source and destination is only one. The number of hops between source and destination differs according to task and the topology followed by the architecture to communicate between subnets or in its own subnet. To calculate the communication delay cores are considered, whereas the hop count considers the number of links that is hops involved in that communication. For all the three architectures, the total number of hops is represented as 'H_T' . In mesh architecture, the total number of hops is the summation of communication setup path hops and data fetch path hops. Communication setup path is comprised of request path from source (S) to destination (D), and acknowledgement path from D to S. Data fetch path is comprised of data path from D to S, and data acknowledgement from S to D. The path from S to D and vice-versa is uniform and so it can be represented as twice the hop count of S to D and twice the hop count of from D to S. Apart from the communication hops, there will be some additional hop requirements in WNoC-CD and WNoC-DDs architectures to update the directories for data synchronization. Unlike mesh architectures, directory based architectures are free from the hops of acknowledgement and they are reliable, efficient to manage the activities of cores. The hop count in WNoC-CD needs more hops compared to proposed model for the Tasks such as 9, 18, 21, 22, 23, 24, and, 25 for updating the directory to maintain data consistency for all above Tasks, as the network is designed with centralized directory. The proposed has the advantage of having wireless router with individual directory to each subnet and thus avoids extra hop counts compared to WNoC-CD and mesh multicasting. For all cases, the hop count for the proposed system is smaller than that for the centralized directory and mesh architectures. In all the three architectures, the return path is from destination to source indicates the accomplished task of communication.

Let's consider Task 1, which has maximum end-to-end communication. In mesh, to communicate between source and destination core it has 10 intermediate hops. Usually in mesh, it should get an acknowledgement to send any information. So, it has double path for source and destination which makes 20 hop count. Similarly, to acknowledge the information is completely received from destination to source is also double which makes 20 hop count and so in total it takes 40 hop count. In WNoC-CD the request to fetch data is up to centralized directory that is 3 hops and then the return path is from destination to source core that is 6 hops which makes the total as 9 hops. In the proposed WNoC-DDs, the request to fetch data is to its individual directory only as the directories are synced that takes 2 hops and then the return path is 5 hops which makes the total as 7 hops.

4.5. Power consumption

Power consumption is one of the major factors that designers consider making the system performance better. For all the three architectures, power consumption for each task is evaluated. To calculate power consumption, primarily we must identify the wired cores, wireless routers, wired and wireless links, directories etc. involved in a task. In general, the wireless links consume more power than wired but it is least considered disadvantage as the wireless interconnects improve scalability [30–33]. In all the three architectures, adaptive XY routing algorithm is used but the power consumption may vary when compared to other as the interconnect topology plays a vital role in establishing communication between source and destination. Previously, it is explained that in mesh architectures it must check the destination core is idle/busy and then only it can fetch the data. Let's consider Task 1 ((0,0.0) - (1,1.8)) for calculating the power consumption in mesh architecture. The total power (P_{tot}) consumption is the summation of P_1, P_2, and P_3. The task includes the power consumption from S to D as P_1 which includes wired links and cores, similarly power consumption from D to S as P_2, and average power of wired links, wired cores as P_3 are considered because XY routing does not follow a single path or fixed route to reach destination. For calculating P_1 and P_2: P_{wr} represents power of wired link that consumes 1 unit, N_{wr} represents number of wired links and in this task, it has 10 links. P_{cwr} represents power of wired core that consumes 3 units, N_{cwr} represents number of wired cores and in this task, it has 11 cores. For calculating P_3: P_{alwr} represents average power of wired links that consumes 5.5 units, P_{canw} represents average power of wired cores that consumes 19.5 units. The total power consumption for Task 1 is mathematically formulated and is shown below.

In Mesh multicasting:

$$P_{tot} = P_1 + P_2 + P_3$$
$$P_1 = (P_{wr}{}^*N_{wr}) + (P_{cwr}{}^*N_{cwr}) = (1^*10) + (3^*11) = 43$$
$$P_2 = (P_{wr}{}^*N_{wr}) + (P_{cwr}{}^*N_{cwr}) = (1^*10) + (3^*11) = 43$$
$$P_3 = P_{alwr} + P_{canw} = 5.5 + 19.5 = 25$$
$$P_{tot} = 43 + 43 + 25 = 111$$

In WNoC-CD, the assumptions for wired links and cores are identical to mesh architecture, however, the average and routing path to reach destination is different. After each task completion, the directory must be updated. The total power consumption is the summation of power consumed from source to directory as P_{sdr} for requesting data, and power consumed by the directory as P_{cdr} to accomplish the given task. For calculating P_{sdr}: P_{awrsn} represents average power consumed by the wired links in a subnet that consumes 2.5 units; P_{cwr} and N_{cwr} are like mesh architecture assumptions but the number of cores vary; P_{cwl} represents power of wireless router core that consumes 3.3 units; P_{wl} represents power of wireless links that consume 1.1 units. For calculating P_{cdr}: P_{dr} represents power of directory that consumes 6 units as it processes additional activities when compared to regular core; P_{cwl} represents power of wireless router and is explained above. For in-subnet tasks, all the three tasks behave as mesh architecture and so they follow a request path and data fetch path to accomplish the task which may increase the power consumption for directory architectures too. The total power consumption for Task 1 is mathematically formulated and is shown below.

In WNoC-CD: (Out-subnet Task 1: (0,0.0) - (1,1.8))

$$P_{tot} = P_{sdr} + P_{cdr}$$
$$P_{sdr} = P_{awrsn} + (P_{cwr}{}^*N_{cwr}) + P_{cwl} + P_{wl} = 2.5 + (3^*2) + 3.3 + 1.1 = 12.9$$
$$P_{cdr} = P_{dr} + P_{cwl} = 6 + 3.3 = 9.3$$
$$P_{tot} = 12.9 + 9.3 = 22.2$$

In WNoC-CD: (For In-Subnet Task 21: (0,0.6) - (0,0.5))

$$\begin{aligned}
&P_{tot} = P_{sdr} + P_{ds}\\
&P_{sdr} = P_{awrsn} + (P_{cwr}{}^{*}N_{cwr}) + P_{cwl} + P_{wl} = 2.5 + (3^{*}3) + 3.3 + 1.1 = 15.9\\
&P_{ds} = P_{awrsn} + (P_{cwr}{}^{*}N_{cwr}) + P_{cwl} = 2.5 + (3^{*}3) + 3.3 = 14.8\\
&P_{tot} = P_{sdr} + P_{ds} = 15.9 + 14.8 = 30.7
\end{aligned}$$

In proposed WNoC-DDs architecture, the introduction of individual directory in every subnet, makes the data update sync easy and reduces hop count as well as traffic compared to centralized directory. Like in centralized directory, the proposed system must update the directory for every task. However, the hop count in proposed architecture is reduced when compared to centralized directory and mesh architectures. The total power consumption is represented by P_{sdd}; P_{ddr} is like P_{dr} in centralized directory that consumes 6 units of power, and is mathematically formulated as below.

In WNoC-DDs: (Out-subnet Task 1: (0,0.0) - (1,1.8))

$$\begin{aligned}
&P_{tot} = \text{Power consumed between source and distributed directoriesrepresented as } P_{sdd}\\
&P_{sdd} = P_{awrsn} + (P_{cwr}{}^{*}N_{cwr}) + P_{ddr} + 3(P_{wl}) = 2.5 + (3^{*}1) + 6 + 3(1.1) = 14.8\\
&P_{tot} = 14.8
\end{aligned}$$

In WNoC-DDs: (For In-Subnet Task 21: (0,0.6) - (0,0.5))

$$\begin{aligned}
&P_{tot} = P_{sdd} + P_{dsddr}\\
&P_{sdd} = P_{awrsn} + (P_{cwr}{}^{*}N_{cwr}) + P_{ddr} + 3\,(P_{wl}) = 2.5 + (3^{*}3) + 6 + 3(1.1) = 20.8\\
&P_{dsddr} = P_{awrsn} + (P_{cwr}{}^{*}N_{cwr}) + P_{ddr} = 2.5 + (3^{*}3) + 6 = 17.5\\
&P_{tot} = P_{sdd} + P_{dsddr} = 20.8 + 17.5 = 38.3
\end{aligned}$$

For in-subnet tasks such as Tasks 21 to 25, the proposed architecture consumes more power when compared to WNoC-CD due to the introduction of individual directory in each subnet. However, on average the hop count is less compared to WNoC-CD and mesh architectures. For the tasks inside subnet, the three architectures must acknowledge back from destination to source with data and thus they consume more power compared to outside subnet tasks. For in-subnet tasks, the proposed WNoC-DDs behaves just like a mesh architecture; however, additional power is required to operate the directories. For out-subnet Tasks 1 to 20, the hop count is drastically reduced compared to other architectures. Thus, we can say the proposed architecture improves the better performance with reduced power consumption on average.

4.6. Average of parameters in percentage

The performance of each parameter such as communication delay, hop count, and power consumption are derived from the three architectures by providing 25 different tasks as workload. The tasks are considered with the scenarios, that has minimum length to maximum length between nodes. The performance of tasks can be observed individually as task wise for communication delay, hop count, and power consumption. However, the overall performance such as average calculation of each parameter gives precise statistics, whether to consider the new proposed architecture is beneficial compared to the other architectures. To find the decrease or improved performance of any parameter, the total column of each architecture is summed initially. The summed column of proposed architecture is subtracted from other architectures individually and finds the reduced difference.

To find the average in percentage, the ratio of reduced difference (proposed architecture subtracted from other architecture) to other individual architecture summed column, and then multiplied by 100. Mathematically, it can be represented as follows:

For calculating average of parameters compared to mesh in % =

$$\frac{\sum_{i=1}^{n=25} Parameter\ of\ Mesh - \sum_{i=1}^{n=25} Parameter\ of\ Proposed}{\sum_{i=1}^{n=25} Parameter\ of\ Mesh} X\ 100 \tag{1}$$

For calculating average of parameters compared to WNoC-CD in % =

$$\frac{\sum_{i=1}^{n=25} Parameter\ of\ WNoC\ with\ CD - \sum_{i=1}^{n=25} Parameter\ of\ Proposed}{\sum_{i=1}^{n=25} Parameter\ of\ WNoC\ with\ CD} X\ 100 \tag{2}$$

Using Eqs. (1) and (2), the average of all parameters in percentage are calculated and therefore the results can be quantified in terms of improved performance. If we know the average, the performance of proposed can be estimated by considering the worst and best scenarios.

5. Results and discussion

Experimental results of all the three architectures are discussed in this section. The performance comparison of communication delay, hop count and power consumption for each individual task is illustrated in Tables 4–6. Table 4 represents the communication delay for all architectures. When the source to destination is directly connected or with no intermediate

Table 4
Communication delay for three different architectures.

Different scenarios	Mesh (multicasting) (ms)	WNoC-CD (ms)	Proposed architecture (ms)
Task 1: (0,0.0) - (1,1.8)	4 × 9 + 40 = 76	4 × 2 + 40 = 48	4 × 1 + 40 = 44
Task 2: (0,0.4) - (1,1.4)	4 × 5 + 40 = 60	4 × 0 + 40 = 40	4 × 0 + 40 = 40
Task 3: (0,0.7) - (1,0.1)	4 × 4 + 40 = 56	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 4: (0,0.3) - (0,1.5)	4 × 4 + 40 = 56	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 5: (1,0.5) - (0,1.2)	4 × 4 + 40 = 56	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 6: (1,0.7) - (0,1.5)	4 × 3 + 40 = 52	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 7: (0,1.0) - (1,0.0)	4 × 5 + 40 = 60	4 × 2 + 40 = 48	4 × 1 + 40 = 44
Task 8: (0,0.8) - (1,1.6)	4 × 3 + 40 = 52	4 × 2 + 40 = 48	4 × 1 + 40 = 44
Task 9: (0,0.7) - (0,1.1)	4 × 0 + 40 = 40	4 × 0 + 40 = 40	4 × 0 + 40 = 40
Task 10: (1,1.5) - (1,0.2)	4 × 3 + 40 = 52	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 11: (0,1.3) - (0,0.1)	4 × 4 + 40 = 56	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 12: (0,1.4) - (1,0.6)	4 × 3 + 40 = 52	4 × 0 + 40 = 40	4 × 0 + 40 = 40
Task 13: (1,0.1) - (1,1.1)	4 × 2 + 40 = 48	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 14: (1,1.2) - (0,0.8)	4 × 3 + 40 = 52	4 × 2 + 40 = 48	4 × 1 + 40 = 44
Task 15: (1,0.6) - (0,1.2)	4 × 1 + 40 = 44	4 × 2 + 40 = 48	4 × 1 + 40 = 44
Task 16: (1,0.4) - (0,1.7)	4 × 6 + 40 = 64	4 × 0 + 40 = 40	4 × 0 + 40 = 40
Task 17: (1,1.3) - (0,1.3)	4 × 2 + 40 = 48	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 18: (0,1.2) - (1,1.0)	4 × 0 + 40 = 40	4 × 0 + 40 = 40	4 × 0 + 40 = 40
Task 19: (0,0.1) - (1,0.7)	4 × 4 + 40 = 56	4 × 1 + 40 = 44	4 × 0 + 40 = 40
Task 20: (1,0.2) - (0,1.6)	4 × 9 + 40 = 76	4 × 2 + 40 = 48	4 × 1 + 40 = 44
Task 21: (0,0.6) - (0,0.5)	4 × 2 + 40 = 48	4 × 2 + 40 = 48	4 × 2 + 40 = 48
Task 22: (1,0.7) - (1,0.8)	4 × 0 + 40 = 40	4 × 0 + 40 = 40	4 × 0 + 40 = 40
Task 23: (0,1.4) - (0,1.2)	4 × 1 + 40 = 44	4 × 1 + 40 = 44	4 × 1 + 40 = 44
Task 24: (1,1.6) - (1,1.2)	4 × 3 + 40 = 52	4 × 3 + 40 = 52	4 × 3 + 40 = 52
Task 25: (0,1.7) - (0,1.1)	4 × 1 + 40 = 44	4 × 1 + 40 = 44	4 × 1 + 40 = 44

Table 5
Hop counts for three different architectures.

Different scenarios	Mesh (multicasting)	WNoC-CD	Proposed architecture
Task 1: (0,0.0) - (1,1.8)	HC = H_T *2 (S to D) + H_T *2 (D to S) = 20 + 20 = 40	HC = H_T (S to Directory) + H_T (D to S) = 3 + 6 = 9	HC = H_T (S to Directory) + H_T (D to S) = 2 + 5 = 7
Task 2: (0,0.4) - (1,1.4)	HC = 12 + 12 = 24	HC = 1 + 2 = 3	HC = 0 + 1 = 1
Task 3: (0,0.7) - (1,0.1)	HC = 10 + 10 = 20	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 4: (0,0.3) - (0,1.5)	HC = 10 + 10 = 20	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 5: (1,0.5) - (0,1.2)	HC = 10 + 10 = 20	HC = 2 + 5 = 7	HC = 1 + 4 = 5
Task 6: (1,0.7) - (0,1.5)	HC = 8 + 8 = 16	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 7: (0,1.0) - (1,0.0)	HC = 12 + 12 = 24	HC = 3 + 6 = 9	HC = 2 + 5 = 7
Task 8: (0,0.8) - (1,1.6)	HC = 8 + 8 = 16	HC = 3 + 6 = 9	HC = 2 + 5 = 7
Task 9: (0,0.7) - (0,1.1)	HC = 2 + 2 = 4	HC = 1 + 1 + 2 = 4	HC = 1 + 1 = 2
Task 10: (1,1.5) - (1,0.2)	HC = 8 + 8 = 16	HC = 2 + 5 = 7	HC = 1 + 4 = 5
Task 11: (0,1.3) - (0,0.1)	HC = 10 + 10 = 20	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 12: (0,1.4) - (1,0.6)	HC = 8 + 8 = 16	HC = 1 + 4 = 5	HC = 0 + 3 = 3
Task 13: (1,0.1) - (1,1.1)	HC = 6 + 6 = 12	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 14: (1,1.2) - (0,0.8)	HC = 8 + 8 = 16	HC = 3 + 6 = 9	HC = 2 + 5 = 7
Task 15: (1,0.6) - (0,1.2)	HC = 4 + 4 = 8	HC = 3 + 6 = 9	HC = 2 + 5 = 7
Task 16: (1,0.4) - (0,1.7)	HC = 14 + 14 = 28	HC = 1 + 3 = 4	HC = 0 + 2 = 2
Task 17: (1,1.3) - (0,1.3)	HC = 6 + 6 = 12	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 18: (0,1.2) - (1,1.0)	HC = 2 + 2 = 4	HC = 1 + 1 + 3 = 5	HC = 1 + 1 = 2
Task 19: (0,0.1) - (1,0.7)	HC = 10 + 10 = 20	HC = 2 + 4 = 6	HC = 1 + 3 = 4
Task 20: (1,0.2) - (0,1.6)	HC = 20 + 20 = 40	HC = 3 + 6 = 9	HC = 2 + 5 = 7
Task 21: (0,0.6) - (0,0.5)	HC = 6 + 6 = 12	HC = 3 + 3 = 1 = 7	HC = 3 + 3 = 6
Task 22: (1,0.7) - (1,0.8)	HC = 2 + 2 = 4	HC = 1 + 1 + 2 = 4	HC = 1 + 1 = 2
Task 23: (0,1.4) - (0,1.2)	HC = 4 + 4 = 8	HC = 2 + 2 + 1 = 5	HC = 2 + 2 = 4
Task 24: (1,1.6) - (1,1.2)	HC = 8 + 8 = 16	HC = 4 + 4 + 1 = 9	HC = 4 + 4 = 8
Task 25: (0,1.7) - (0,1.1)	HC = 4 + 4 = 8	HC = 2 + 2 + 1 = 5	HC = 2 + 2 = 4

cores, and in-subnet Tasks such as 21 to 25 the communication delay is identical in all architectures as they basically exhibit the function of mesh architecture. Tasks such as 9 and 18 with no intermediate cores behave as mesh and so the delay is similar. For the tasks, where all the three architectures behave as mesh are highlighted in Table 4.

The average communication delay for all 25 tasks is calculated by using Eqs. (1) and (2), and is shown in Fig. 3. From the average of all workload, it can be noticed that the proposed architecture reduces the communication delay up to 20.54% and 5.40% when compared to mesh and WNoC-CD architectures respectively.

Table 5 illustrates the hop count of all the three architectures. When the hop distance is one between two cores, or when the core is directly connected to a core then the hop count is identical in all architectures as they basically exhibit

Table 6
Power consumption for three different architectures.

Different scenarios	Mesh (multicasting) (mW)	WNoC-CD (mW)	Proposed architecture (mW)
Task 1: (0,0.0) - (1,1.8)	$P_{tot} = P_1 + P_2 + P_3$ $P_{tot} = 43 + 43 + 25 = 111$	$P_{tot} = P_{sdr} + P_{cdr}$ $P_{tot} = 12.9 + 9.3 = 22.2$	$P_{tot} = P_{sdd}$ $P_{tot} = 14.8$
Task 2: (0,0.4) - (1,1.4)	$P_1 = 24$, $P_2 = 24$, $P_3 = 25$, $P_{tot} = 73$	$P_{sdr} = 6.9$, $P_{cdr} = 9.3$ $P_{tot} = 16.2$	$P_{tot} = 11.8$
Task 3: (0,0.7) - (1,0.1)	$P_1 = 23$, $P_2 = 23$, $P_3 = 25$ $P_{tot} = 71$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 4: (0,0.3) - (0,1.5)	$P_1 = 23$, $P_2 = 23$, $P_3 = 25$ $P_{tot} = 71$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 5: (1,0.5) - (0,1.2)	$P_1 = 23$, $P_2 = 23$, $P_3 = 25$ $P_{tot} = 71$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 6: (1,0.7) - (0,1.5)	$P_1 = 19$, $P_2 = 19$, $P_3 = 25$ $P_{tot} = 63$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 7: (0,1.0) - (1,0.0)	$P_1 = 27$, $P_2 = 27$, $P_3 = 25$ $P_{tot} = 79$	$P_{sdr} = 12.9$, $P_{cdr} = 9.3$ $P_{tot} = 22.2$	$P_{tot} = 14.8$
Task 8: (0,0.8) - (1,1.6)	$P_1 = 19$, $P_2 = 19$, $P_3 = 25$ $P_{tot} = 63$	$P_{sdr} = 12.9$, $P_{cdr} = 9.3$ $P_{tot} = 22.2$	$P_{tot} = 14.8$
Task 9: (0,0.7) - (0,1.1)	$P_1 = 7$, $P_2 = 7$, $P_3 = 25$ $P_{tot} = 39$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 10: (1,1.5) - (1,0.2)	$P_1 = 19$, $P_2 = 19$, $P_3 = 25$ $P_{tot} = 63$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 11: (0,1.3) - (0,0.1)	$P_1 = 23$, $P_2 = 23$, $P_3 = 25$ $P_{tot} = 71$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 12: (0,1.4) - (1,0.6)	$P_1 = 19$, $P_2 = 19$, $P_3 = 25$ $P_{tot} = 63$	$P_{sdr} = 6.9$, $P_{cdr} = 9.3$ $P_{tot} = 16.2$	$P_{tot} = 11.8$
Task 13: (1,0.1) - (1,1.1)	$P_1 = 15$, $P_2 = 15$, $P_3 = 25$ $P_{tot} = 55$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 14: (1,1.2) - (0,0.8)	$P_1 = 19$, $P_2 = 19$, $P_3 = 25$ $P_{tot} = 63$	$P_{sdr} = 12.9$, $P_{cdr} = 9.3$ $P_{tot} = 22.2$	$P_{tot} = 14.8$
Task 15: (1,0.6) - (0,1.2)	$P_1 = 11$, $P_2 = 11$, $P_3 = 25$ $P_{tot} = 47$	$P_{sdr} = 12.9$, $P_{cdr} = 9.3$ $P_{tot} = 22.2$	$P_{tot} = 14.8$
Task 16: (1,0.4) - (0,1.7)	$P_1 = 31$, $P_2 = 31$, $P_3 = 25$ $P_{tot} = 87$	$P_{sdr} = 6.9$, $P_{cdr} = 9.3$ $P_{tot} = 16.2$	$P_{tot} = 11.8$
Task 17: (1,1.3) - (0,1.3)	$P_1 = 15$, $P_2 = 15$, $P_3 = 25$ $P_{tot} = 55$	$P_{sdr} = 9.9$, $P_{cdr} = 9.3$ $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 18: (0,1.2) - (1,1.0)	$P_1 = 7$, $P_2 = 7$, $P_3 = 25$ $P_{tot} = 39$	$P_{sdr} = 12.9$, $P_{cdr} = 9.3$ Ptot = 22.2	$P_{tot} = 11.8$
Task 19: (0,0.1) - (1,0.7)	$P_1 = 23$, $P_2 = 23$, $P_3 = 25$ $P_{tot} = 71$	$P_{sdr} = 9.9$, Pcdr = 9.3 $P_{tot} = 19.2$	$P_{tot} = 11.8$
Task 20: (1,0.2) - (0,1.6)	$P_1 = 43$, $P_2 = 43$, $P_3 = 25$ $P_{tot} = 111$	$P_{sdr} = 12.9$, $P_{cdr} = 9.3$ $P_{tot} = 22.2$	$P_{tot} = 14.8$
Task 21: (0,0.6) - (0,0.5)	$P_1 = 15$, $P_2 = 15$, $P_3 = 25$ $P_{tot} = 55$	$P_{tot} = P_{sdr} + P_{ds}$ $P_{tot} = 15.9 + 14.8 = 30.7$	$P_{tot} = P_{sdd} + P_{dsddr}$ $P_{tot} = 20.8 + 17.5 = 38.3$
Task 22: (1,0.7) - (1,0.8)	$P_1 = 7$, $P_2 = 7$, $P_3 = 25$ $P_{tot} = 39$	$P_{sdr} = 12.9$, $P_{ds} = 8.5$ $P_{tot} = 21.4$	$P_{tot} = 17.8 + 8.5 = 26.3$
Task 23: (0,1.4) - (0,1.2)	$P_1 = 11$, $P_2 = 11$, $P_3 = 25$ Ptot = 47	$P_{sdr} = 12.9$, $P_{ds} = 11.8$ $P_{tot} = 24.7$	$Pt_{ot} = 17.8 + 14.5 = 32.3$
Task 24: (1,1.6) - (1,1.2)	$P_1 = 19$, $P_2 = 19$, $P_3 = 25$ $P_{tot} = 63$	$P_{sdr} = 18.9$, $P_{ds} = 17.8$ $P_{tot} = 36.7$	$P_{tot} = 23.8 + 20.5 = 44.3$
Task 25: (0,1.7) - (0,1.1)	$P_1 = 11$, $P_2 = 11$, $P_3 = 25$ $P_{tot} = 47$	$P_{sdr} = 12.9$, $P_{ds} = 11.8$ $P_{tot} = 24.7$	$P_{tot} = 17.8 + 14.5 = 32.3$

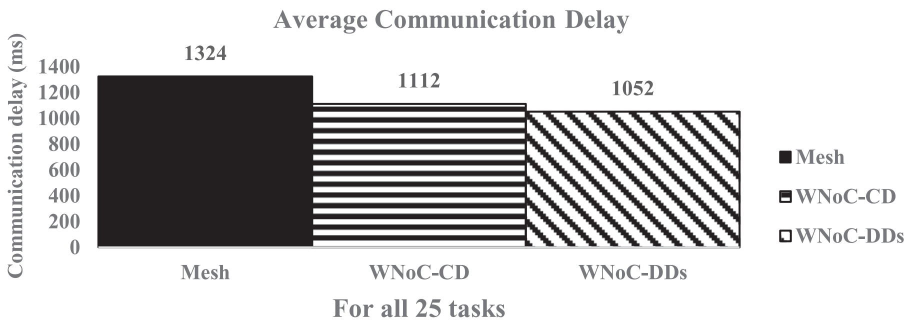


Fig. 3. Average communication delay for different architectures.

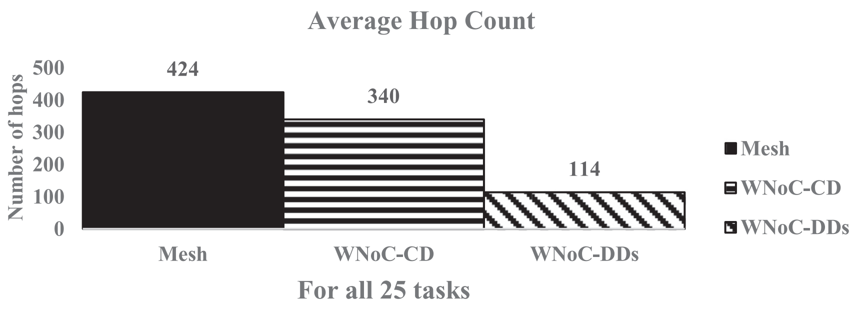


Fig. 4. Average hop count for different architectures.

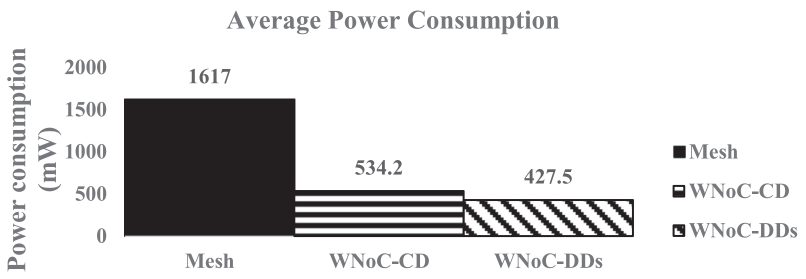


Fig. 5. Average power consumption for different architectures.

the function of mesh architecture. One hop count tasks are highlighted in Table 5. WNoC-CD needs extra hops (based on task) to update the centralized directory for maintaining data sync.

The average hop count for all 25 tasks is calculated by using Eqs. (1) and (2), and is shown in Fig. 4. From the average of all workload, it can be noticed that the proposed architecture reduces the hop count to 73.11% and 29.19% when compared to mesh and WNoC-CD architectures respectively.

Table 6 illustrates the power consumption of all the three architectures. The results prove that the introduction of individual directory in proposed architecture reduces the power consumption significantly.

The individual directory in proposed architecture skips the excessive hops, cores when compared to mesh and WNoC-CD architectures. WNoC-CD needs extra hops (based on task) to update the centralized directory for maintaining data sync. The data synchronization is faster when compared to WNoC-CD and it promises the distributed directories layout can also reduce scalability issues. The traffic is better controlled when compared to other architectures. However, the power consumption for in-subnet tasks escalated due to the individual directory for a subnet. Generally, the directory consumes twice the power of a regular core, which is 6 unit. For quick observation, in-subnet Tasks 21 to 25 are highlighted in Table 6.

Fig. 5 illustrates the average power consumption for all 25 tasks and is calculated by using Eqs. (1) and (2). From the average of all workload, it can be noticed that the proposed architecture reduces the power consumption up to 73.56% and 19.97% when compared to mesh and WNoC-CD architectures respectively. The proposed broadcasts the accomplished task information to other subnets for maintaining the data sync among directories/subnets. The proposed distributed directories (P_{ddr}) take less hops, and less nodes involved in accomplishing a task, than a centralized directory (P_{cdr}) because of directory sync in each subnet.

6. Conclusions

Performance of modern Network-on-Chip architectures depends on communication latency, hop count, and power consumption. If the communication setup-time is quick, then the system performance should be better. In this work, we introduce a WNoC architecture with distributed directories (WNoC-DDs) to improve the performance to power ratio. A directory allows the tasks to execute faster by providing adaptive minimal routing path to reach the destination node. VisualSim Architect is used to model and simulate the architectures by using synthetic workload. It is observed that the distributed directories significantly improve the performance of WNoC architecture, which supports the adaptability of WNoC-DDs to larger networks. The experimental results suggest that the proposed WNoC-DDs architecture improves the overall performance. The improvement in the proposed architecture is majorly due to the introduction of directories and their predominant services to their subnets. In WNoC-CD, the directory handles all cores to maintain efficient communication and the traffic is larger as the tasks completion is sequential. In the proposed WNoC-DDs, individual subnets can operate simultaneously if/as the cores acquire the required data from its own subnet. As the individual directory maintains/tracks the status of other directories, it would take less time for processing without or any further queries for the required data. Finally, each of the distributed directories can control substantial number of cores compared to centralized directory.

The 3D NoC architecture has advantages such as stacking multiple active layers, low power dissipation, and reduced hop count. However, temperature concerns due to high power density and heavier switching are the major challenging issues. We plan to investigate the impact of combining 3D routers with 3D processor architectures in our next endeavor.

References

[1] Hu J, Marculescu R. Energy-aware mapping for tile-based NoC architectures under performance constraints. In: Proceedings of the 2003 Asia and South Pacific design automation conference. ACM; 2003 January, p. 233–239.
[2] Taylor MB, Lee W, Amarasinghe S, Agarwal A. Scalar operand networks: on-chip interconnect for ILP in partitioned architectures. In: High-performance computer architecture, 2003. HPCA-9 2003. Proceedings. The ninth international symposium. IEEE; 2003. p. 341–53.
[3] Held J, Bautista J, Koehl S. From a few cores to many: a tera-scale computing research overview, White paper. Intel; 2006.
[4] Zhu K, Ding Y. Research on low power scheduling of heterogeneous multi core mission based on genetic algorithm. In: Measuring technology and mechatronics automation (ICMTMA). IEEE. 2017 9th International conference; 2017 Jan 14, p. 219–223.
[5] Bezerra GB. Energy consumption in networks on chip: efficiency and scaling. The University of New Mexico; 2012.
[6] Liu J, Mahapatra NR. The role of interconnects in the performance scalability of multicore architectures. In: SOC Conference, 2008 IEEE international. IEEE; 2008 Sep 17. p. 21–4.
[7] Khan O, Lis M, Sinangil Y, Devadas S. Dcc: a dependable cache coherence multicore architecture. IEEE Comput Archit Lett 2011;10(1):12–15.
[8] Daya BK, Chen CH, Subramanian S, Kwon WC, Park S, Krishna T, Holt J, Chandrakasan AP, Peh LS. SCORPIO: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering. ACM SIGARCH Comput Archit News 2014 Oct 16;42(3):25–36.
[9] Xu TC, Leppänen V, Forsell M. Exploration of a heterogeneous concentrated-sparse on-chip interconnect for energy efficient multicore architecture. In: Computer and information technology (CIT), 2014 IEEE international conference on. IEEE; 2014 Sep 11. p. 204–11.
[10] Deb S, Sah SP, Cosic M, Chang K, Yu X, Heo D, Ganguly A, Belzer B, Pande PP. Design of an energy efficient CMOS compatible NoC architecture with millimeter-wave wireless interconnects. IEEE Trans. Comput 2012 Sep 19;99(1):1.
[11] Park D, Vaidya A, Kumar A, Azimi M. MoDe-X: microarchitecture of a layout-aware modular decoupled crossbar for on-chip interconnects. IEEE Trans Comput 2014;63(3 (March)):622–36.
[12] Chawade SD, Gaikwad MA, Patrikar RM. Review of XY routing algorithm for Network-on-Chip architecture. Int J Comput Appl 2012;43(April):975–8887.
[13] Asaduzzaman A, Chidella KK, Vardha D. An energy-efficient directory based multicore architecture with wireless routers to minimize the communication latency. IEEE Trans Parallel Distrib Syst 2017 Feb 1;28(2):374–85.
[14] Zoni D, Flich J, Fornaciari W. Cutbuf: buffer management and router design for traffic mixing in vnet-based nocs. IEEE Trans. Parallel Distrib. Syst. 2016 Jun 1;27(6):1603–16.
[15] Karsten M, Berger DS, Schmitt J. Traffic-driven implicit buffer management-delay differentiation without traffic contracts. In: Teletraffic congress (ITC 28), 2016 28th International, vol. 1. IEEE; 2016 Sep 12. p. 44–52.
[16] Liu G., Schmidt T., Dömer R., Dingankar A., Kirkpatrick D. Optimizing thread-to-core mapping on manycore platforms with distributed Tag Directories. In Design automation conference (ASP-DAC). IEEE. 2015 20th Asia and South Pacific 2015 Jan 19; p. 429–434.
[17] Ros A, Acacio ME, García JM. A scalable organization for distributed directories. J Syst Archit 2010 Mar 31;56(2):77–87.
[18] Wang X, Schulzrinne H, Kandlur D, Verma D. Measurement and analysis of LDAP performance. ACM SIGMETRICS Perform Eval Rev 2000 Jun 18;28(1):156–65.
[19] Deshpande S, Ravale P, Apte S. Cache coherence in centralized shared memory and distributed shared memory architectures. Int J Comput Sci Eng (IJCSE) 2010:39–44.
[20] Cerutti I, Behredin AM, Andriolli N, Ladouceur OL, Castoldi P. Ring versus bus topology: a network performance comparison of photonic integrated NoC. In: Transparent optical networks (ICTON), 2016 18th International conference on. IEEE; 2016 Jul 10. p. 1–4.
[21] Pintus P, Gambini F, Faralli S, Di Pasquale F, Cerutti I, Andriolli N. Ring versus bus: a theoretical and experimental comparison of photonic integrated NoC. J Lightwave Technol 2015 Dec 1;33(23):4870–7.
[22] Park D, Vaidya A, Kumar A, Azimi M. MoDe-X: microarchitecture of a layout-aware modular decoupled crossbar for on-chip interconnects. IEEE Trans Comput 2014;63(3 (March)):622–36.
[23] Feero BS, Pande PP. Networks-on-chip in a three-dimensional environment: a performance evaluation. IEEE Trans Comput 2009;58(1 (January)):32–45.
[24] Agyeman MO. A study of optimization techniques for 3d networks-on-chip architectures for low power and high performance applications. Int J Comput Appl 2015;121(6) Jan 1.
[25] Wu T, Kuo GS. An analytical model for centralized service discovery architecture in wireless networks. In: Vehicular technology conference, 2006. VTC-2006 Fall; 2006 IEEE 64th; 2006. p. 1–5. Sep 25.
[26] Lenoski DE, Weber WD. Scalable shared-memory multiprocessing. Elsevier; 2014. Jun 28.
[27] Lenoski D, Laudon J, Gharachorloo K, Weber WD, Gupta A, Hennessy J, Horowitz M, Lam MS. The stanford dash multiprocessor. Computer 1992;25(3 (March)):63–79.
[28] Chen X, Chen W, Yang S, Lu Z, Wang Z. DASH: a duplication-aware flash cache architecture in virtualization environment. In: Parallel and distributed systems (ICPADS), 2014 20th IEEE International conference. IEEE; 2014 Dec 16, p. 842–847.
[29] VisualSim Architect. Mirabilis Design. http://mirabilisdesign.com/new/visualsim/; 2016 [accessed on 10/15/17]
[30] Fang J, Lu J, She C. Research on topology and policy for low power consumption of Network-on-Chip with multicore processors. In: Computational science and computational intelligence (CSCI); IEEE; 2015 International conference on; 2015 Dec 7, p. 621–625.
[31] Biagetti G, Crippa P, Curzi A, Orcioni S, Turchetti C. ToLHnet: a low-complexity protocol for mixed wired and wireless low-rate control networks. In: Education and research conference (EDERC). IEEE. 2014 6th European embedded design; 2014 Sep 1, p. 177–181.
[32] Mondal HK, Deb S. An energy efficient wireless Network-on-Chip using power-gated transceivers. In: System-on-Chip conference (SOCC); 2014 27th IEEE International; 2014 Sep 2, p. 243–248.
[33] Deb S, Ganguly A, Pande PP, Belzer B, Heo D. Wireless NoC as interconnection backbone for multicore chips: promises and challenges. IEEE J Emerging Sel Top Circuits Syst 2012;2(2 (June)):228–39.

Kishore K. Chidella received M.Tech degree in Embedded Systems from JNTU, India and B.E in ECE from Anna University, India. He is currently working towards the Ph.D. degree in Embedded Systems at Wichita State University, USA. His research interests include embedded systems, sensors monitoring, high-performance computing, and low-power computer architecture. He has published several articles out of his research work.

Abu Asaduzzaman is currently Associate Professor of Computer Engineering at Wichita State University. He received research grants from NSF KS EPSCoR, NVIDIA, and NetApp. His research interests include computer architecture, high performance computing, and embedded systems. He has authored more than 80 peer-reviewed journal and conference articles out of his research work. He is a member of IEEE and ASEE.